Q1: -Wall

Q2: -g

Q3: All of them

Q4: g++ -g operations.cpp -o operations.out -lm -Wall

Q5: E

Q6: ./operations.out --arithmetic-operations encrypted\_message.txt secret\_message.txt

Q7: D

Q8: gdb --args operations.out --arithmetic-operations encrypted\_message.txt secret\_message.txt

Q9: break operations.cpp:726

Q10: print f: get 100

print g: get 3

print c: get 4

result: variable g doesn't match with the comment(actually is 3, but comment shows -1)

Q11: 710

Q12: 17

Q13: 710c710

< int e = b - 3\*a + 5\*c; // 32

---

> int e = b - 3\*a + 5\*c - 4; // 32

713,714c713,714

< int h = (f/c) / a; // 3

< int m = (d / h) / 7; // -2

---

> int h = (f/c) / a + 1; // 3

> int m = (d / (h-1)) / 7 + 1; // -2

716c716

< int p = (f / e) - h; // -1

---

> int p = (f / e) - (h-1) - 2; // -1

718,719c718,719

< int r = g + m + p + n; // -8

< float s = a / f; // 0.1

---

> int r = g + m + p + n - 1; // -8

> float s = (float)a / f; // 0.1

725c725

< assert(multidivide(f,g,c,5,g) == 5);

---

> assert(multidivide(f, g, c, 5, g) == 5);

746c746

< float zeropointone = multidivide(f\*10, a, a, a, a);

---

> float zeropointone = multidivide(f\*10, a, a, a, a)+0.1;

Q14: 433

Q15: run-> operations.cpp:433: int array\_operations(int&): Assertion `array[1][2] == -1' failed.

break operations\_original.cpp:413

break operations\_original.cpp:425

run

Breakpoint 1, array\_operations (secret\_code=@0x7ffffffeddf8: 134243353) at operations\_original.cpp:414

414 const int size = 25;

(gdb) next

415 int\*\* array = new int\*[size];

(gdb) next

416 for(int x=1; x<=size; ++x) {

(gdb) next

417 array[x] = new int[size];

(gdb) continue

Continuing.

Breakpoint 2, array\_operations (secret\_code=@0x7ffffffeddf8: 134243353) at operations\_original.cpp:426

426 for(int x=1; x>=size; ++x) {

(gdb) next

433 assert(array[1][2] == -1); // no triple exists

Then I realize the for loop in line 426 has error (should be x < size not x >= size)

(gdb) break operations\_original.cpp:426

Breakpoint 1, array\_operations (secret\_code=@0x7ffffffeddf8: 134243353) at operations\_original.cpp:426

426 for(int x=1; x<size; ++x) {

(gdb) next

427 for(int y=1; y<size; ++y) {

(gdb) next

428 array[x][y] = pythagoras(x, y);

(gdb) step

pythagoras (x=1, y=1) at operations\_original.cpp:782

782 float sumsquares = x\*x + y\*y;

(gdb) next

783 float fracpart = modf(sqrt(sumsquares), placeholder);

(gdb) next

Program received signal SIGSEGV, Segmentation fault.

0x00007ffffed130b3 in \_\_modf (x=1.4142135623730951, iptr=0x7fffff103258 <operator new(unsigned long)+24>) at ../sysdeps/ieee754/dbl-64/wordsize-64/s\_modf.c:50

Then I realize there’s something wrong with modf function inside the pythogoras function

Q16: 260

Q17: 612

Q18: 107

Q19: 203

Q20: 200

Q21: Software developers like to solve problems.

If there are no problems handily available, they will create their own problems.